EXHIBIT 5

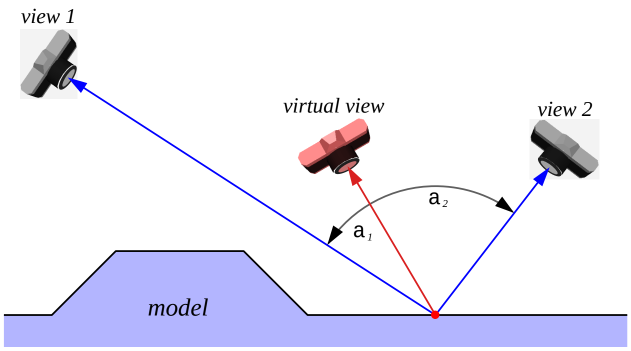

Figure 12: *The weighting function used in view-dependent texture mapping. The pixel in the virtual view corresponding to the point on the model is assigned a weighted average of the corresponding pixels in actual views 1 and 2. The weights w_1 and w_2 are inversely inversely proportional to the magnitude of angles a_1 and a_2. Alternately, more sophisticated weighting functions based on expected foreshortening and image resampling can be used.*

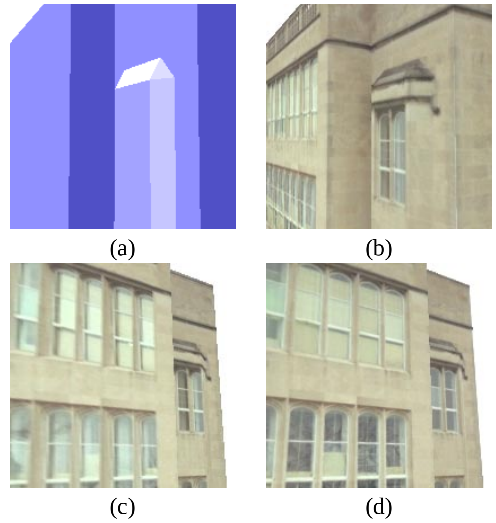

Figure 13: *View-dependent texture mapping.* **(a)** *A detail view of the high school model.* **(b)** *A rendering of the model from the same position using view-dependent texture mapping. Note that although the model does not capture the slightly recessed windows, the windows appear properly recessed because the texture map is sampled primarily from a photograph which viewed the windows from approximately the same direction.* **(c)** *The same piece of the model viewed from a different angle, using the same texture map as in (b). Since the texture is not selected from an image that viewed the model from approximately the same angle, the recessed windows appear unnatural.* **(d)** *A more natural result obtained by using view-dependent texture mapping. Since the angle of view in (d) is different than in (b), a different composition of original images is used to texture-map the model.*

address this, the pixel weights are ramped down near the boundary of the projected images. Although this method does not guarantee smooth transitions in all cases, we have found that it eliminates most artifacts in renderings and animations arising from such seams.

If an original photograph features an unwanted car, tourist, or other object in front of the architecture of interest, the unwanted object will be projected onto the surface of the model. To prevent this from happening, the user may mask out the object by painting over the obstruction with a reserved color. The rendering algorithm will then set the weights for any pixels corresponding to the masked regions to zero, and decrease the weights of the pixels near the boundary as before to minimize seams. Any regions in the composite image which are occluded in every projected image are filled in using the hole-filling method from [23].

In the discussion so far, projected image weights are computed at every pixel of every projected rendering. Since the weighting function is smooth (though not constant) across flat surfaces, it is not generally not necessary to compute it for every pixel of every face of the model. For example, using a single weight for each face of the model, computed at the face's center, produces acceptable results. By coarsely subdividing large faces, the results are visually indistinguishable from the case where a unique weight is computed for every pixel. Importantly, this technique suggests a real-time implementation of view-dependent texture mapping using a texture-mapping graphics pipeline to render the projected views, and α-channel blending to composite them.

For complex models where most images are entirely occluded for the typical view, it can be very inefficient to project every original photograph to the novel viewpoint. Some efficient techniques to determine such visibility *a priori* in architectural scenes through spatial partitioning are presented in [18].

4 Model-Based Stereopsis

The modeling system described in Section 2 allows the user to create a basic model of a scene, but in general the scene will have additional geometric detail (such as friezes and cornices) not captured in the model. In this section we present a new method of recovering such additional geometric detail automatically through stereo correspondence, which we call *model-based* stereo. Model-based stereo differs from traditional stereo in that it measures how the actual scene deviates from the approximate model, rather than trying to measure the structure of the scene without any prior information. The model serves to place the images into a common frame of reference that makes the stereo correspondence possible even for images taken from relatively far apart. The stereo correspondence information can then be used to render novel views of the scene using image-based rendering techniques.

As in traditional stereo, given two images (which we call the *key* and *offset*), model-based stereo computes the associated depth map for the key image by determining corresponding points in the key and offset images. Like many stereo algorithms, our method is *correlation-based*, in that it attempts to determine the corresponding point in the offset image by comparing small pixel neighborhoods around the points. As such, correlation-based stereo algorithms generally require the neighborhood of each point in the key image to resemble the neighborhood of its corresponding point in the offset image.

The problem we face is that when the key and offset images are taken from relatively far apart, as is the case for our modeling method, corresponding pixel neighborhoods can be foreshortened very differently. In Figs. 14(a) and (c), pixel neighborhoods toward the right of the key image are foreshortened horizontally by nearly a factor of four in the offset image.

The key observation in model-based stereo is that even though two images of the same scene may appear very different, they appear similar after being projected onto an approximate model of the scene. In particular, projecting the offset image onto the model and viewing it from the position of the key image produces what we call the *warped offset* image, which appears very similar to the key image. The geometrically detailed scene in Fig. 14 was modeled as two flat surfaces with our modeling program, which also determined the relative camera positions. As expected, the warped offset image (Fig. 14(b)) exhibits the same pattern of foreshortening as the key image.

In model-based stereo, pixel neighborhoods are compared between the key and warped offset images rather than the key and off-